Massachusetts Institute of Technology

Artificial Intelligence Laboratory

AI MEMO 858
(replaces #164A)

(revision 2)
24 May, 1976

The Text-Justifier
TJ6

by

Joseph D. Cohen

Abstract

This memo, intended as both a reference and user's manual describes the text-justifying program TJ6, which compiles a neat output document from a sloppy input manuscript. TJ6 can justify and fill text; automatically number pages and figures; control page format and indentation; underline, superscript, and subscript; print a table of contents; etc.

Work reported herein was conducted at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the laboratory's artificial intelligence research is provided in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research Contract number N00014-75-C-0643

Introduction

TJ6 is a compiler which translates a sloppy input manuscript into a neat output document. It can justify and fill text; automatically number pages and figures; control page format and indentation; underline, super and subscript; print a table of contents; and do a few other strange things.

TJ6 tries to be a fast, simple program which meets most documenting needs, rather than a large, complicated, relatively slow program like PUB, which tries to meet all documenting needs. In order to achieve speed, small size, and simplicity, TJ6 has naturally sacrificed the potential for many of PUB's capabilities. TJ6 does not have: columnar output, macros, footnotes, indices, block structure, conditional commands, or arithmetic or string calculations.

This memo tries to be both a reference manual and a user's manual. Towards the first end, this document is intended to be complete, even to the point of describing some of TJ6's inner workings. Towards the second end, this memo describes things in a manner which an experienced programmer might think too simple or verbose. This is intended to be useful to someone who has no programming or TJ6 experience. (If that description fits you, first have a look at section 20, Sample Input.)

Conventions

This memo also includes examples of the use of most commands, which should be a help not only to the novice, but to someone trying to find a mysterious bug in his TJ6 program. Text which appears in

```
fixed width roman type
```

is an example of input text. Text which appears in

micrograмma type

is an example of output text.

An underbar in examples of input text denotes a space, e.g.,

```
foo_
```

means "foo ".

Commands

TJ6 input is divided into text and command lines. Command lines are lines which start with a period. (Except that the line after SPREAD, HEADER, HE1, and HE2 is always interpreted as text even if it starts with a period.) The name of the command is the word following the period.
Text lines:

```
This is a text line.
⊃.This is also a text line.
```

Command lines:

```
.INDENT 5
.GUN
```

(Even though there's no such command as GUN, the second line is still a command line because it starts with a period.) Only the first six letters of the command name are significant. Any extra stuff on command lines will be ignored. Some characters: →, ←, ∈, ↑, ↓, ↔, ∧, ∞, π, and ⊃ have special meaning in text lines. (They are dealt with in section 16, In-text Character Commands.)

Many commands are followed by an argument, which will be denoted below by:
<integer> a decimal integer;
<file name> a standard ITS file name like "DSK:TJ6;TJ6 ORDER";
<character> any ASCII character;
<decimal> an integer or a psuedo floating point number, with up to three places to the right of the decimal, e.g., 5, 6., 7.089.
Any other argument abbreviations should be read in context.

Many commands which have no explicit inverse have the inverted effect if given the argument of 0.

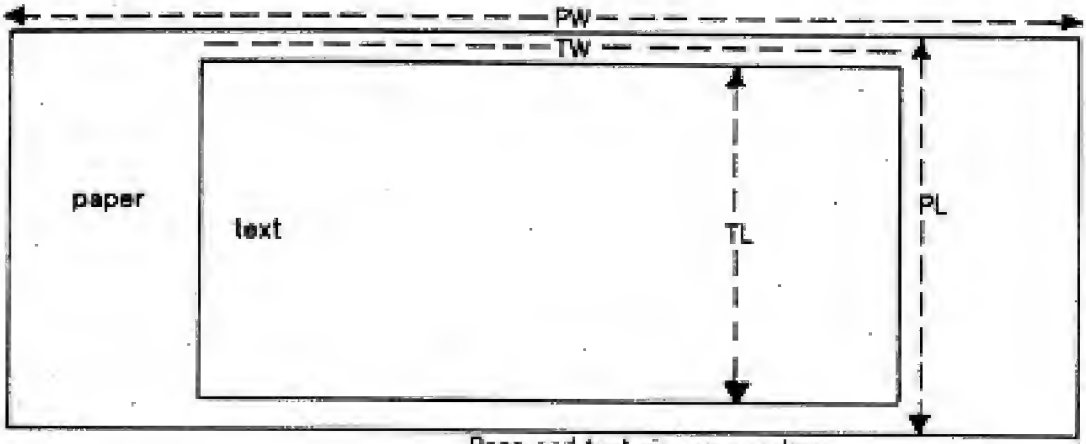
Many commands have abbreviations. These are listed on the line following the command.

Each command is described as follows:

Long form <argument(s)>
B indicates this command causes a break. (Text will not be filled across it.)
I indicates command can be inverted with an argument of 0.
abbreviation(s)
Description

Page Format

These commands control which area of the page text may appear in, and where the heading appears. (See section 10, Headings.) Commands which give additional control within the defined text area (such as by indenting) are covered in section 8, Horizontal Margins.



Page and text size parameters

Whenever one of these parameters is changed, TJ6 readjusts the margins so that the proportional size of left to right margin and top to bottom margin stays the same. (Those proportions can be changed with the SIDM and TOPM commands.) All of these commands also move the heading.

PL <number> B
Set paper length to <number> lines.

PLINCH <decimal> B
Like PL but argument in inches.

PW <number> B
Set paper width to <number> "spacing units".

PWINCH <decimal> B
Like PW, but argument in inches.

TL <number> B
Set the text length to <number> lines.

TLINCH <decimal> B
Like TL, but argument in inches.

TW <number> B
Set text width to <number> "spacing units". (See section 7, Spaces and Tabs.) (A spacing unit is usually the largest character in FONT 0.)

TWINCH <decimal> B
Like TW but argument in inches. For example,

```
.TWINCH 3.5
"If the past were permitted to weigh on its conscience,
every nation would be compelled to commit hara-kiri."
```

is output as:

"If the past were permitted to weigh on its conscience, every nation would be compelled to commit hara-kiri."

Note how the percentage of margin on each side is kept constant.

SINCH <a,b,c,d,e> B

Set PWINCH, TWINCH, PLINCH, TLINCH, and left margin in inches, respectively to <a>, <b>, <c>, <d>, and <e>. (All the arguments are decimal.) This memo was produced with a

```
.SINCH 8.5, 6.5, 11, 9, 1
```

The default for the XGP is SINCH 8.5,6,11,8,1.25. (When not using the XGP, the defaults are: PW 85, TW 73, PL 66, and TL 50. For a typewriter with a character width of 1/10 inch and line height of 1/6 inch, this works out to SINCH 8.5,7.3,11,8.333,0.6.)

TOPM <number> B

Make the top margin <number> percent of the total.

```
.PLINCH 11
.TLINCH 9
.TOPM 60
```

will make the top margin 1.2 inches (60 percent of 11-9) and the bottom margin 0.8 inches.

SIDM <number> B

Make left margin <number> percent of the available margin space. The setting of SIDM, normally 50%, effects the appearance of lines whose size has been change by TW, TWINCH, or RTW. SIDM is kept constant through changes in TW.

Filling, Adjusting, Centering, and Aligning

This section deals with the basic "justification" operations of TJ6--the commands which make the output neat, even though the input is ragged. ADJUST and FILL also affect the treatment of punctuation marks and redundant spaces in the input text. This is described in section 12, Periods and Punctuation.

ADJUST B
AD

Adjust the output text. (Align its left and right edges by inserting spaces within the line.) For example, the input text,

```
.ADJUST
"Lilies that fester smell far worse than weeds:
both India and Japan seem to be spiritually sicker,
more estranged from a living faith than the West.
They are at opposite ends of the Asian spectrum,
whose centre is occupied by the vastness of China,
one of the world's oldest cultures;
yet it proved even less resistant against the impact
of a materialistic ideology."
```

is output as:

"Lilies that fester smell far worse than weeds: both India and Japan seem to be spiritually sicker, more estranged from a living faith than the West. They are at opposite ends of the Asian spectrum, whose centre is occupied by the vastness of China, one of the world's oldest cultures; yet it proved even less resistant against the impact of a materialistic ideology."

FILL B
FI

Approximately align right margin by not printing a word which would go beyond it. (The default mode.) With .FILL instead of .ADJUST, the previous example becomes:

"Lilies that fester smell far worse than weeds: both India and Japan seem to be spiritually sicker, more estranged from a living faith than the West. They are at opposite ends of the Asian spectrum, whose centre is occupied by the vastness of China, one of the world's oldest cultures; yet it proved even less resistant against the impact of a materialistic ideology."

Filling vs. Adjusting

There is some argument as to whether filled or adjusted text looks better. While the even right margin of adjusted text can be easier on the eye, there are some problems which can be introduced by the addition of space in the middle of the line. Typically, all the adjusting is done by putting additional spaces between words. This can cause "rivers", where several adjacent lines have extra space in about the same column. This problem is especially bad with short lines, as in newspaper columns. TJ6 tries to avoid rivers by trying to put space between letters before putting space between words. TJ6 figures out how much space has to be inserted to adjust the line to the right margin; puts as much space as possible between letters, while keeping the space between letters equal; and then puts the remainder of the space between words. This can lead to another problem -- the intercharacter space is large compared to the interword space. The MIT Press is avoiding these problems by making many publications filled, not adjusted.

NOFILL B
NF

Don't fill text. (This is not the equivalent of PUB's VERBATIM. NOFILL simply causes TJ6 to begin a new output line for every new input line.) Tabs and leading spaces in NOFILL

and FILL modes occupy the same amount of space. (See section 7, Spaces and Tabs.) A NOFILL line which sticks out past the right margin will generate an error message, but will be printed anyway.

NVRADJ BZ

Never adjust. Ignore future adjust commands.

CENTER B
CE

Center the very next input line.

```
.CENTER
Winnie the Pooh
```

Winnie the Pooh

RIGHT B
RI

Right justify the next input line.

```
.RIGHT
Winnie the Pooh
```

Winnie the Pooh

SPREAD B

The next line is three strings, the first will be left justified, the second centered, and the third right-justified. The strings are separated by a separator defined as the very first character on the next line.

```
.SPREAD
/Winnie/the/Pooh/
```

Winnie the Pooh

Breaking

When TJ6 is filling or adjusting, it runs together lines of input text in the output. This section deals with what causes TJ6 to break, i.e., to begin a new output line.

TJ6 reads characters into an internal buffer until it finds a word boundary which sticks past the right margin. Then it goes back through the buffer until it finds a word boundary which doesn't stick past the margin. The output text is broken at that word boundary. A word boundary can be:

- a space;
- a carriage return line feed;
- a hypen;
- a hidden hypen. (See HYPCHR below.)

In addition, the user may cause the output text to be broken by:

- a breaking command (one which is followed by a B in this memo);
- a blank line or lines;
- a line with leading spaces and/or tabs;
- the BREAK command.

HYPHEN Z

Don't allow - to break a line. For example, the following text will ordinarily be broken at the hyphen in "Lao-Tse":

"The nation which held fast for ... millenia to the teaching of Confucius, Lao-Tse and the Buddha, succumbed to the atheistic doctrine formulated by the son of a German lawyer, and has become the most accomplished robot state this side of science fiction."

When preceded, by a .HYPHEN, the text will be broken at the space preceeding the hyphen.

"The nation which held fast for ... millenia to the teaching of Confucius, Lao-Tse and the Buddha, succumbed to the atheistic doctrine formulated by the son of a German lawyer, and has become the most accomplished robot state this side of science fiction."

HYPCHR <character> B

Define the "hidden hyphen" character which will not be printed unless used to break a word, in which case it will be printed as -. This can be used as a super-stupid syllabification, or in a special case when there is an extraordinarily long word, which, when broken at the word boundary, looks awful. HYPCHR with no argument turns off the current HYPCHR.

```
.HYPCHR ¬
To look to Asia for mys¬tic en¬light¬en¬ment and spir¬itual
guid¬ance has be¬come as much of an a¬nach¬ron¬ism as to think of
A¬mer¬ica as the Wild West.
```

To look to Asia for mystic enlightenment and spiritual guidance has become as much of an anachronism as to think of America as the Wild West.

BREAK B
BR

Don't fill input text across this command.

```
Don't fill text
.BREAK
across the BREAK command.
```

Don't fill text
across the BREAK command.

CRBR B

Make every carriage return cause a break. (Opposite of CRSP.) This command can be used as a sort of compromise between FILL and NOFILL mode, where lines are filled, (or adjusted) but every time you begin a new input line, TJ6 begins a new output line.

```
.TWINCH 2.5
.CRBR
This memo, TJ6 Text Justifier, is divided into the following sections:
Introduction,
Commands,
Page Format,
etc.
```

This memo, TJ6 Text Justifier, is divided in the following sections:
Introduction,
Commands,
Page Format,
etc.

CRSP

When not in NOFILL mode, carriage returns cause spaces, instead of breaks. (The default. Opposite of CRBR.)

SPBR B

Leading spaces or tabs (see section 7, Spaces and Tabs), cause a break. (The default. Opposite of SPSP.) Tabs and spaces that cause breaks will appear in the output text. (They will not be compacted by SPCOMP. See section 12, Periods and Punctuation.)

SPSP

Opposite of SPBR. This command is useful for allowing text which will be indented in the output to also be indented in the input, without worrying about where that text will be broken.

```
.SPSP
As Arthur Koestler has remarked,
.INDENT 4
     "Asians have a tendency to lay the blame for this decline
     on the soul-destroying influence of the West,
     and Western intellectuals are inclined to accept the blame."
```

As Arthur Koestler has remarked,

"Asians have a tendency to lay the blame for this decline on the soul-destroying influence of the West, and Western intellectuals are inclined to accept the blame."

SPSP works this way because the leading spaces in the indented lines are compacted in ADJUST mode. To do this in FILL mode use the SPCOMP command. (See section 12, Periods and Punctuation.)

Blank Lines

This section deals with how to leave extra spacing (blank lines) in a page. Extra spacing can be left with the commands described, or by leaving blank lines in the input text.

TJ6 is not as clever about blank input lines as it is about the SPACE command. Consecutive blank input lines may be split across a page boundary in the output. When in NOFILL mode, blank input lines appear as blank output lines. This is the default condition in FILL or ADJUST mode, but may be turned off and on with CRCOMP and CRRETA.

SINGLE B
SS

Single space the output text. (The default spacing.)

```
.SINGLE
These two lines are
single spaced.
```
These two lines are
single spaced.

DOUBLE B
DS

Double space output text.

```
.DOUBLE
These two lines are
double spaced.
```
These two lines are

double spaced.

SINGLE and DOUBLE take effect when the next line is output, not immediately.

```
These
.DOUBLE
four
.SINGLE
little
lines.
```
These
four

little
lines.

SPACE <number> B
SP

Insert <number> contiguous blank lines in the output text. (When DOUBLE spacing, leave 2*<number> blank lines.) (If not enough space at end of page, does a PAGE. SPACE is slightly cleverer than blank lines in the input text, in that SPACE 1 is ignored at the top of page.)

```
Blank
.SPACE
Line
Blank
```

Line

But when double spacing,

```
. DOUBLE
Blank
. SPACE
Line
Blank


Line
```

(Compare this with the next example.)

ASP <number> B

Absolute space. Just like SPACE, but inserts <number> blank lines in the output text regardless of whether one is SINGLE or DOUBLE spacing.

```
. DOUBLE
Blank
. ASP
Line
Blank


Line
```

(Compare this with the previous example.)

FIGURE <number>

Leave <number> contiguous blank lines at the next available break. If there is room on the current output page, leave <number> blank lines at the next break; if no room on the current output page, at the top of the next page. (The argument is an absolute number of lines and is not effected by SINGLE or DOUBLE spacing.) FIGURE is really iterative, i.e., if there is not space at the top of the next output page, (due to an earlier FIGURE), leave space at the top of the next available page.

```
This command
. FIGURE 3
causes three consecutive blank lines to be left somewhere, without
interrupting the output stream at the command.
This command causes three consecutive blank lines to be left somewhere, without



interrupting the output stream at the command.
```

Another way of leaving room for illustrations is with the BLOCK command. (See section 9, Pagination.)

FINCH <decimal>

Like FIGURE, but argument in inches.

```
Just as in the previous example,
. FINCH 0.5
there should be a one half inch figure
somewhere in the middle of this line.
Just as in the previous example, there should be a one half inch figure somewhere in



the middle of this line.
```

CRRETA B

Retain redundant carriage returns. (The default. Opposite of CRCOMP.)

```
.CRETA
Blank

Line
Blank

Line
```

(Compare this with the next example.)

CRCOMP B

When not in NOFILL mode, compact redundant carriage returns. (Opposite of CRRETA.)

```
.CRCOMP
Blank

Line
```

prints as:

```
Blank Line
```

The total effect of carriage returns is determined not only by CRRETA and CRCOMP, but also by CRBR and CRSP. (See section 5, Breaking.)

Leading Spaces and Tabstops

When using variable width fonts, TJ6 goes to some extra trouble in order to:

1) Make things which are aligned with tabs in TECO also align in TJ6;

```
.NOFILL
123456789
111111  tabstop
WWWWWW  tabstop
123456789
iiiiii          tabstop
WWWWWWW         tabstop
```

There is a tab after the i's and W's, so the word "tabstop" is aligned in both input and output.

2) Make <number> spaces at the beginning of a line equivalent to INDENT <number>.

```
_One leading space is the same as
.INDENT 1
indenting one.
```

__One leading space is the same as
indenting one.

3) and since in TECO and ITS generally, there is a tabstop every 8 spaces, TJ6 makes INDENT <number*8> equivalent to <number> tabs;

To accomplish this, the size of a space at the beginning of a line; the size of indentation units; and the size of the TW and PW unit, referred to as the "spacing unit", will not necessarily be equal to the size of a space in any font. (In particular, note that the size of a leading space is usually different from the size of a space in the middle of a line. The former is a spacing unit, the latter is a space in the appropriate font.) Compare

__________six leading spaces with
____six imbedded spaces.

Leading spaces and tabs are tabs which appear at the beginning of a line with no other characters before them on the line. A line with in-text characters preceeding spaces does not have leading spaces, e.g.,

```
_This line has a leading space.
→ ←This line does not.
```

__This line has a leading space.
_This line does not.

The default size of the spacing unit is the size of the widest character in FONT 0. In the micrographma font, W is the widest character, so if FONT 0 is 25VMIC, W is the spacing unit. Then,

__One leading space is the same width as
.INDENT 1
indenting one, and also the same width as a
W.

A tab stop is equal to 8 spacing units. (Of course, if font 0 is fixed width, the spacing unit is identical to a FONT 0 space, and a tab equals eight spaces.) Leading tabs and spaces are all printed in spacing units. The following commands all use the spacing unit as the unit of their argument: INDENT, UNDENT, OF, RINDEN, RTW, PW, and TW. (PW and TW actually are internally translated to TWINCH and PWINCH, so that changing the size of the spacing unit will not change the text width or paper width.)

The default is chosen on the assumption that all the characters to the left of tabstops are in FONT 0. If this is not true, you should probably use:

TABFNT <number>

where <number> is the only or largest font which will always be to the left of tabs, or <number>=-1 if any font can be to the left of tabs. Make the spacing unit the largest character in FONT <number>, or if <number>=-1, make the spacing unit the largest character in all fonts. The default is TABFNT 0. For example, the largest character in both 25VMIC and 25VG is W, but in 25VMIC it's 28 XGP dots wide, while in 25VG it's 23 wide.

```
.FONT 0 25VG
.FONT 2 25VMIC
.TABFNT 2
        Tab in 25VMIC
.TABFNT 0
        Tab in 25VG
_____Tab in 25VMIC
_____Tab in 25VG
```

SPW <number>

Set the size of the spacing unit to <number> XGP dots. For example, in 25FR, all characters are 17 XGP dots wide.

```
WWWWWW
__A leading space is normally 28 dots wide, but with
.SPW 17
_it's 17 dots wide.
WWWWWW
```

One of the results of all this is that if the TABFNT is variable width, tabstops will be much larger than expected, and there will be a gross difference between leading and imbedded spaces. (Look at the previous examples.) The spacing unit with 25VMIC is 100% larger than with 25FGB. The large spacing unit size is necessary to guarantee alignment of tabstops. If there can be 7 microgramma W's to the left of a tabstop, than a tab obviously in the worst case should be 8 W's wide. Most of the time, you won't have the worst case (all wide characters within a tabstop), and you can get away with a SPW 16 (width of 25FG). This will result in indentations and tabs whose size is intuitively more expected.

Horizontal Margins

This section deals with changing the margins of text within the area declared by the page format commands (PW and TW).

INDENT <number> B
IN

Indent the text left margin by <number> "spacing units". (See section 7, Spaces and Tabs.)

```
.INDENT 2
Indents two spaces.
```

  Indents two spaces.

RINDEN <number> B
RIN

Increment the indentation by <number>. <Number> may be negative.

```
.RINDEN 1
Indent one.
.RINDEN 1
Indent one more.
```

  Indent one.
    Indent one more.

UNDENT <number> B
UN

Start the next line <number> "spacing units" left of the current INDENTation. You may not UNDENT past the left margin. Undent should appear immediately before the line to be UNDENTed. (See INDENT and section 7, Spaces and Tabs.)

```
Cox's lecture begins,
.UNDENT -8
"To understand our labor laws one must first understand the genius
of our labor movement."
```

Cox's lecture begins,
"To understand our labor laws one must first understand the genius of our labor movement."

OF <number> B

Increase the indentation by <number> after the next line is output. (Just like
.RIN <number>
.UN <number>.)

```
.OF 4
"Although negotiation carries its own compulsions, few 'voluntary'
agreements are executed in the absence of economic power."
```

"Although negotiation carries its own compulsions, few 'voluntary' agreements are executed in the absence of economic power."

RLINE <number> B
RTW

Decrease the text width by <number> without affecting where the header is printed. This command is really a way of simultaneously indenting both left and right margin. The amount of additional indentation which will appear on either side is determined by SIDM. (See section 3, Page Format.)

```
"The only functions of law, it is thought, are:
.RTW 4
(1) to protect the formation of unions, so that employees may exert
sufficient bargaining power to solve the problems of workers in an
```

```
industrial society; and
.BREAK
(2) to provide the framework for negotiations between the employer and
employees..."
```

"The only functions of law, it is thought, are:

(1) to protect the formation of unions, so that employees may exert sufficient bargaining power to solve the problems of workers in an industrial society; and

(2) to provide the framework for negotiations between the employer and employees..."

Note that <number> is the total increment to the indentation of both sides. Since, in this memo, SIDM is 50, the total indentation of 4 is divided equally into 2 for the left margin and 2 for the right.

Pagination

This section deals with forcing TJ6 to start a new output page, and with the page number which appears in the heading. (Also see section 10, Headings.) The page number is normally an integer, but can be made into a "decimal", consisting of <chapter>.<page> through the CHAP command. The first output page is page number 1.

PAGE <number> B
PA

Start a new output page and increment page number by <number>. If no argument increment by one. Note that there is a difference between PAGE <n> and n PAGEs. The former starts a new page with a number <n> higher than the current page, while the latter actually prints n-1 blank pages.

BLOCK <number> B

If there is not room for <number> lines of text on this page, begin a new output page. BLOCK takes into account whether you are SINGLE or DOUBLE spacing. I.E., BLOCK n when DOUBLE spacing leaves twice as much space as when SINGLE spacing.

ABLOCK <number> B

If there are less than <number> lines left on this page, begin a new output page. ABLOCK does not take into account SINGLE or DOUBLE spacing. ABLOCK is useful as a substitute for FIGURE, when you want a diagram to appear at that spot in the text, e.g.,

```
.ABLOCK 3
.SPACE 3
```

(The ABLOCK prevents the SPACE from being split by a page boundary.)

SPAGE <number>

Set page number at the next page break to <number>. (A page break is when TJ6 has too much text to put on a page, or a new page is forced by a PAGE, BLOCK, or other command.) (SPAGE with no argument makes the next page number one.)

EPAGE <number>
EP

Increment the page number by <number> at the next page boundary.

PD B

Odd page force. Do a PAGE, and if the next page number would not be odd, do an EPAGE to make it odd.

PV B

Even page force. Works like PD.

CHAP <number> B

Does an SPAGE 1, and if <number>≠0, causes <number> to be printed before page number in headings (see section 10, Headings). If there is no argument, the chapter number is incremented. This page was started with:

```
.CHAP 9
.PAGE
→Pagination←
```

Headings

This section deals with headings -- where and when they are printed. The default TJ6 heading is "PAGE " followed by the page number, all printed in font 0 (see section 14, XGP), and appearing one line from the top of the page, aligned with the right margin (set by TW, TWINCH, etc.) The heading is not normally printed on the first page (page 1). Even though TJ6 has a default heading, it's a good idea to always define a heading line, as this gives TJ6 an accurate idea of the heading width. (The width is calculated when the heading is defined, but is only roughly estimated for the deafult heading, since TJ6 doesn't know in advance what font 0 will look like. Therefore, headings should always be defined after the fonts are given.)

HEADER B
HEADIN
HE

The very next input line will be printed by TJ6 as the heading, followed immediately by the page number. (TJ6 won't insert a space between the HEADER and the page number, so make sure that you do.) In-text character commands (see section 16) can appear as input in HEADER lines. For instance,

```
.HEADER
→Important memo←␣
```

is output as:

Important memo 1776

but, without the final space

```
.HEADER
→Important memo←
```

Important memo1776

Another common mistake is to try to put a command line between the HEADER command line and the header. The very next line after .HEADER will be intrepreted as the desired heading.

HE1 B

Make the header a SPREAD line with the page number in the middle, and the very next input line on the left. The center field of the heading is the page number, and the right field is defined with HE2. This memo's heading was generated with

```
.HE1
→TJ6 Text Justifier
```

HE2 B

Make the header a SPREAD line with the page number in the middle, and the very next input line on the right. This section of the memo starts with

```
.CHAP 10
.HE2
Headings←
.PAGE
```

Note the interaction between HE1 and HE2. The begin underline (→) at the start of the HE1 field is finished with an end underline (←) at the end of the HE2 field, so that the whole heading, including page number, is underscored.

NOHEAD BZ

Don't print a heading line. (NOHEAD 0 turns it back on.)

PHP1 Z

Print header on page one. (The default is no heading on page 1 — PHP1 0.)

THESIS B

Print the header inside of the text margins, instead of 2 lines from the top of the page. (Make header appear where the first text line normally would appear.)

HEADPW BZ

Align header with full page width (set by PW rather that TW). For instance, the heading on this page would be

```
. HEADPW
```

Sections and Table of Contents

This section deals with TJ6's cheap table of contents feature, and some commands affecting it. Since TJ6 is a one-pass compiler, the user has to run TJ6 twice to get a finished table of contents. (Two-pass compilers like PUB don't have this problem.) The first time, TJ6 is run with the /C switch. (See section 17, Console Operation.) On that run, TJ6 outputs a file, suitable for TJ6 input, which contains section titles and page numbers. When TJ6 is run on that file, a neat looking table of contents will result. The source file for this memo is TJ6;TJ6MEM >. The DDT command line

```
:TJ6 TJ6MEM /C
```

results in the file TJ6;TJ6MEM CONTEN, which looks like this:

```
.C TJ6-generated Contents
.Spread
/→Introduction←//1.1/
.Spread
/Conventions//1.1/
.Spread
/→Commands←//2.1/
.Spread
/→Page Format←//3.1/
.Spread
/PL <number> →B←//3.1/
etc.
```

When fed back to TJ6, TJ6MEM CONTEN results in the table of contents for this memo. (Actually, TJ6MEM > inserts the file TJ6MEM CONTEN, as well as the file TJ6MEM INDEX, which is merely the CONTEN file modified with a simple TECO alphabetizing command.)

SECTIO B

SECT

The next output line is a section title which will be used when TJ6 outputs a table of contents. (See /C.) SECT also does a BLOCK 2, so that the section title and first line will not appear on seperate pages. (See SBLOCK.) This paragraph of this memo starts with

```
.sect
.of 8
SECTIO B
```

The appropriate line in the CONTENts file is

```
.Spread
/SECTIO →B←//11.1/
```

which finally appears in the table of contents (page 2) as

SECTIO B 11.1

SBLOCK <number>

Set the size of the .BLOCK implied by .SECT to <number>. (The default is SBLOCK 2, so that a section title and the first line of the section will appear on the same page.)

STERM <character>

Set the .SPREAD separator used by TJ6 in writing a table of contents. This command is useful when a / must appear in the section title. For instance

```
.STERM |
.SECTIO
/C
```

appears in the CONTENts file as

```
.Spread
|/C|17.1|
```

but without the .STERM |, would appear as

```
//C//17.1/
```

which would print as

C

Periods and Punctuation

The handling of periods and punctuation is one area in which TJ6 tries harder and has more hair than PUB. When adjusting, TJ6 tries to put two spaces after each of the following characters at the end of a sentence or clause: , ? ! ; and : When justifying, TJ6 will also try to add space at the end of a sentence before adding space between words.

TJ6's algorithm for determining whether one of . ? ! : ; ends a clause or sentence is, given the following character sequence, with . standing for all the punctuation characters:

AB. CD,

the . is the end of a clause or sentence if C is a carriage return or line feed; otherwise, this is not the end of a clause or sentence if:
B is a number; or
B is a letter and A is a space, tab, one of .?!:,; or
C is not a space, tab, carriage return or line feed, or one of)]"; or
C is one or more of)]" and D is not a space, tab, carriage return, or line feed.

In cases like 1AB.) , the two spaces and justifying spaces are inserted after the). TJ6 will determine that the following do end clauses:

```
End. End; "End." ("End.") End"])
```
End. End; "End." ("End.") End"])

and the following do not:

```
No. end 1.414 !n. J. Cohen .no no!)))o
```
No.end 1.414 !n. J. Cohen .no no!)))o,

but beware

```
Mr. Cohen
```
Mr. Cohen

which TJ6 thinks is the end of a sentence.

When in ADJUST mode, TJ6 removes any extra spaces between words in order to make it easier to decide where to put in justifying spaces. Therefore, in adjust mode, the following all have the same affect as one space: tab, tabs, spaces, or combinations thereof.

```
One space  Two          Tab
```
One space Two Tab.

PERIOD <number>,<character> B

Turns off special treatment of <character> if <number>=0, else turns it on. <Character> can be any of .?!:; PERIOD with no arguments turns off special treatment of all of .?!:; When special treatment is turned off, TJ6 will no longer try to insert two spaces after that character.

HALFAD B

Fill the text (see FILL), but flush more than 1 space or tab in a row (SPCOMP), and insert two spaces after any .?!: which ends a clause or sentence. This is a fill mode for people who are sloppy with punctuation.

```
.HALFAD
See the perceptron! Run, perceptron, run! Both
previous sentences have only one space at the end.
```
See the perceptron! Run, perceptron, run! Both previous sentences have only one space at the end.

SPCOMP

When not in NOFILL mode, compact redundant spaces. ADJUST and HALFAD do an automatic SPCOMP. Spaces or tabs which cause breaks will not be compacted. (Opposite of SPRETA.)

SPRETA

Retain redundant spaces. NOFILL and FILL do an automatic SPRETA. (Opposite of SPCOMP.)

File Commands

These commands give TJ6 the means to access other files. The VERSE command and ∞ (see section 16, In-text Character Commands) allow for insertion of file names.

INSRT <file name>

Insert <file name> into the current input file. INSRTs may not be nested more than 7 deep. The source file for this memo is TJ6;TJ6MEM >, which contains all sections but Contents, Error Messages, and Index, which are inserted with

```
.INSRT TJ6;TJ6MEM CONTEN
.INSRT TJ6;ERROR >
.INSRT TJ6;TJ6MEM INDEX
```

APPEND <file name>
AP

Append file <file name> to the input file here.

END B

Ignore the rest of this file. (A file need not end with this command. END is useful when there is stuff at the end of the file that one doesn't want printed. This is useful for putting comments at the end of a file.)

```
...  which proves all the points set forth in this thesis.
.END
Not by a long shot!!!
```

... which proves all the points set forth in this thesis.

VERSE

Insert into the current TJ6 version number into the input stream. (This is mostly useful as a command for debugging TJ6.)

```
This memo was produced using
.VERSE
```

This memo was produced using NNTJ6 25

(The TJ6 source file is stored on the TJ6 directory. The portion of VERSE before the period is the file name of the source. The number after the period represents the number of patches which have been made (number of bugs killed) since last reassembling. The impure binary is on the SYSBIN directory, and the pure binary on the SYS directory.)

XGP

If you intend to use the XGP, there are special TJ6 commands which take advantage of XGP and SCRIMP features such as: choice of fonts, super and subscripts, underscoring. In order to take advantage of these features, you must use TJ6's "XGP" command. This also results in an output file in a strange looking format.

XGP B

Either this command or the /X switch (see section 17, Console Operation) must be used to activate all XGP-type features. If the output device is DSK, TJ6 will output a file intended for printing on the XGP with a default second file name of XGP, and will pay attention to other XGP-only commands and in-text character commands: FONT, SQUISH, VSP, NORMAL, NOSPEC, SCRIPT, BLADJU, PSCORE, ∈, ↑, ↓, and ↔. All of these commands will be ignored unless preceeded by XGP or the /X switch is used. Since XGP be ignored unless it appears before all input text, it's a good idea to have this be the very first line of the input file. (Since the output device must be DSK, you can cause TJ6 to ignore the XGP command by outputting to TTY, AI, etc.) The output file created by TJ6 for the XGP is designed to be used by AI ITS' SCRIMP program, and contains coded data for the XGP, as well as the output text. (The XGP character interpretation is described in AI Working Paper Number 72, XGP Font Catalog.)

FONT <number> <file name> B

Causes the font file <file name> to be associated with font (character set) number <number>. When the output file is finally printed on the XGP, it will appear in the designated fonts. (The font file must be in KST format. See Working Paper #78 if you want to design your own font.) TJ6 has six fonts numbered 0 through 5, and starts printing in font 0. Fonts are switched with the ∈ in-text command. (See section 16, In-text Character Commands.) FONT is ignored if not preceeded by /X switch or XGP command and must appear before text. FONT 0 is special in that it is normally used to determine the size of tabs and leading spaces, (see section 7,) and all fonts are assumed to be the same height. (I.E., although TJ6 understands variable width fonts, it assumes all lines of text are equal in height. This results in a page which is too long if the page contains lots of super or subscripts, or lots of text in a font which is taller than FONT 0.) The default FONT 0 is 25FG. There is no default for higher fonts. A good place to have the FONT specifications is immediately after the XGP command. The source file for this memo starts with

```
.XGP
.FONT 0 25vg
.FONT 1 25fr
.FONT 2 25vmic
.FONT 3 25vri
```

SQUISH

This causes a ;SQUISH to appear on the XGP command page. ;SQUISH causes the XGP program to minimize the number of font character rasters shipped to the XGP's PDP11 and stored there. SCRIMP squishes by reading the entire file before sending fonts to the PDP11. It then sends only the rasters for characters which appear in the TJ6'ed file. (Normally, SCRIMP sends to the PDP11 and stores there the raster for every character in all the FONTs you used. Because of the limited size of the PDP11's memory, it sometimes does not have room for all those rasters.) SQUISH is useful when SCRIMP types out a "CHARACTERS LOST" error message.

VSP <number> B

Set XGP interline spacing to <number> scan lines. This command must appear before all text. VSP is normally 6, i.e., 1/32 inch.

NORMAL <number>

If <number>=0, or no argument, don't print quoted characters on XGP in normal mode. (The default is normalized characters -- NORMAL 1.) Certain characters: ■ (null), λ (backspace), Y (tab), S (linefeed), ± (formfeed), ⊕ (carriage return), and ∫ (rubout), are interpreted by SCRIMP and the XGP as commands, unless they are quoted to the XGP (in which case they are simply printed as characters). Whenever one of these characters is quoted in the TJ6 source file, TJ6 will in turn quote it to the XGP. The quoting of formatting characters to force them to be printing characters is "normalization". (To avoid this, use NORMAL 0.)

```
W⊃<backspace>|
.NORMAL 0
W⊃<backspace>|
WA|
W
```

The quoted backspace is usually output as a lambda. In non-normal mode, the quoted backspace is passed on to the XGP as a backspace.

NOSPEC B

Don't output an XGP command page. In XGP mode, TJ6 normally outputs a page of SCRIMP commands which looks like

```
;SIZE 11
;VSP 6
;TOPMAR 0
;BOTMAR 0
;LFTMAR 0
;SKIP 1
;KSET FONTS;25VG KST,FONTS;25FR KST,FONTS;25VMIC KST,,,
AI:JDC;NNTJ6 10.0
```

By making the left margin 0, (SIDM 0), and using NOSPEC, the user may, for his own perverse reasons, output his own SCRIMP command page.

SCRIPT <number>

Set the number of XGP lines that ↑ (superscript) or ↓ (subscript) will raise or lower the baseline. (See section 16, In-text Character Commands.) (The default is SCRIPT 12, 1/16 inch.)

```
.SCRIPT 12
E=MC↑12
.SCRIPT 6
MC↑6↔=E
E=MC12 MC6=E
```

BLADJU <number>

Sets the baseline to <number> amount. + is up. BLADJU 0 is the same as ↔. <number> must be between -64 and +63.

```
E=MC
.BLADJU 8
8
E=MC 8
```

PSCORE <number>

Sets the position of underscoring to <number> scan lines below font base line. (- is up.) Underscoring done with the → and ← in-text command characters is normally 3 XGP scan lines below the font base line (PSCORE 3).

```
→Ordinary underline (PSCORE 3)←
.PSCORE 0
→Raised underscore←
```

Ordinary underline (PSCORE 3) Raised underscore

Miscellaneous Commands

COMMEN
C

Treat the entire command line as a comment.

```
Winnie
.COMMEN I like Piglet lots.
the Pooh
Winnie the Pooh
```

(But remember that excess on any command line is also treated as a comment.)

```
Winnie
.UNDENT -2 I also like Eeyore and Rabbit.
the Pooh
Winnie
   the Pooh
```

DUMMY <character> <number> B

If no second argument or <number>≠0, defines a character which will be treated a letter internally, but print as a space on output. If <number>=0, turns off DUMMYing for that character. (There is no default DUMMY character. There may be more than one dummy.)

```
.DUMMY @
1@2@@4@@@@.
1 2  4   .
```

QUOTE <character> B

<Character> becomes the TJ6 quoting character, which is used in input text to quote the following character. (Also see section 16, In-text Character Commands.) To turn off use QUOTE with no argument. (The default is QUOTE ⊃. (control Q))

```
.QUOTE @
@@@→
@→
```

TRANS ab,cd, . . .

On input, translate character <a> to <b>, <c> to <d>, etc. Characters are not translated within command lines. To undo, "translate" the character back to itself.

```
.TRANS WD,eu,lt,sc
Welsh
.TRANS WW,ee,ll,ss
Welsh
Dutch
Welsh
```

(Since translation occurs before interpretation of in-text commands, TRANS can be used in the same way as PUB's TURN ON FOR.)

GENNUM <number> <counternumber>

Increment GENNUM counter number <counternumber> by <number> and insert the incremented number. If no second argument, use counter 1. If no arguments, increment counter 1 by 1. TJ6 has eight GENNUM counters numbered 1-8.

```
.GENNUM
.GENNUM 3
.GENNUM 2 1
1 4 B
```

To get GENNUMs without contiguous spaces, use the π in-text command. (See section 16, In-text Character Commands.)

GENSET <number> <counternumber>

Set GENNUM counter number <counternumber> to <number>-1, so that when the next GENNUM or π is used, <number> will appear. If no second argument, use counter 1. If no arguments, reset counter number 1 to 0.

```
.GENSET 1 1
.GENNUM
1
```

ULFONT <number> B

If not using the XGP, underscore all characters in font number <number>. When not using the XGP, this command is handy for causing an italic XGP font to be replaced by underscoring. (Only one font at a time can be the ULFONT.)

```
.ULFONT 3
Berman's book, ε3Talks on American Lawε0, ...
```

appears on the XGP as,

Berman's book, *Talks on American Law*, ...

but appears elsewhere as,

Berman's book, Talks on American Law, ...

In-text Character Commands

These are characters which appear in text lines, but have the effect of commands. (All in-text commands can also appear in HEADING lines.) (They cannot be turned on and off as in PUB, but since translation (see TRANS, section 15, Miscellaneous Commands) occurs before in-text commands are interpreted, TRANS can be used as a sort of TURN ON FOR.)

→ (control Y)

Begin underscore.

```
→The House at Pooh Corner←
```

The House at Pooh Corner

Don't forget to turn off underscoring with the ← command. TJ6 underscores by printing a backspace and underscore following each underscored character, except on the XGP where the XGP underscore commands are used. (The location of the XGP underscore is controlled by the PSCORE command. See section 14, XGP Commands.) This means that on the XGP, an underscore made with the → and ← characters is different than one made by putting backspaces and underbars in the input text. Compare:

```
→The House at Pooh Corner.←
The House at Pooh Corner.
The House at Pooh Corner.
The  House  at  Pooh  Corner.
```

Use of → and ← is preferred because it is: easier to type in; easier to edit; takes less space in the source file; and looks better on the XGP than underbars.

← (control X)

End underscore. Used to terminate scope of → begin underscore command.

ε<n> (control F)

Select font number <n>. In this file fonts 0 through 3 are respectively: 25VG, 25FG, 25VMIC, and 25VRI.

```
Fonts ε0zero, ε1one, ε2two, and ε3three.
```

Fonts zero, one, two, and *three*.

π<n> (control G)

Insert a generated number. This command has exactly the same effect as GENNUM 1 <n>, or with no argument, is the same as GENNUM 1 1. It is more flexible than GENNUM, however, since a GENNUMed number is surrounded by spaces.

```
Figure 1.π
```

Figure 1.1

If a GENNUM had been used here, there would have been a space after the period.

```
Figure 1.
.GENNUM
```

Figure 1. 1

↑ (control K)

Superscript. (Raise baseline by SCRIPT.) The baseline will continue to be raised untill changed by another ↑ or ↓, or reset by a ↔.

```
A↓2↑↑n↔=B↓n↑↑2
```

$A_2^{n}=B_n^{2}$

↓ (control A)

Subscript. (Lower baseline by SCRIPT.) See previous example.

↔ (control W)

Reset baseline to 0. See previous example.

^ (control D)

Insert date in format: Month day, Year.

```
Today is ^.
```

Today is May 17, 1978.

∞ (control N)

Insert current input file name in standard ITS format DEV:SYSNAM;FNAM1 FNAM2.

```
The source for this section of the TJ6 memo is ∞.
```

The source for this section of the TJ6 memo is DSK:TJ6;TJBMEM 60.

⊃ (control Q)

Default QUOTE character. (See QUOTE.)

```
∞
⊃∞
DSK:TJ6;TJBMEM 60
∞
```

Console Operation

TJ6 console operation is basically like that of most other compilers. The program is started from DDT in the normal way. It takes two file names as arguments, first the output name, then the input name. It compiles the output file, typing out any errors it finds, and when it is done, TJ6 kills itself.
In this section *italics* will be used for all computer generated typeout.
The following command strings will both compile the file JDC;TEST MEMO from the source file JDC;TEST >:

```
TJ6↑/
NNTJ6 11.1
_JDC;TEST MEMO_JDC;TEST >
```

or simply

```
:TJ6 TEST
```

Command Line Switches

There are some special "commands" which can be given as part of the original TJ6 command line, and which will be ignored when part of a source file. They are "command line switches" and consist of a / followed by a character.

/C

Output a table of contents intended for input to TJ6. Make the output default second file name CONTEN instead of MEMO. The table of contents will be a series of .SPREADs, with the left field the .SECTIO section titles, and the right field the page number. The separator will be / or a character defined by STERM. (See section 11, Sections and Table of Contents.)

Using the TJ6 contents feature requires running two passes of TJ6, the one to produce the table of contents, the other to produce the memo. This memo has both a table of contents and a crude index which are both INSRTed into the orginal file in order to produce the final memo. The DDT command string

```
:TJ6 TJ6;TJ6MEM /X/C
```

uses TJ6;TJ6MEM >, and outputs only a table of contents as the file TJ6;TJ6MEM CONTEN. TJ6MEM CONTEN is then alphabetized in TECO to produce the index file TJ6;TJ6MEM INDEX.

```
:TJ6 TJ6;TJ6MEM /X
```

uses TJ6;TJ6MEM > to produce the final XGPable memo as the file TJ6;TJ6MEM XGP.

When using the /C command, make sure that TJ6 knows what device the memo will eventually be output on, otherwise the page numbers may be incorrect. If the memo is eventually going to the XGP, make sure to use the /X or XGP command. If not using the XGP, use neither command, or specify an output device other than DSK.

/X

Output for the XGP. This command is ignored unless output is to DSK. This switch is exactly the same as the XGP command.

/T

Take input from the teletype before reading the specified file. TJ6 will type out the prompt character > and treat input from the teletype as though it appears at the beginning of the specified source file. TJ6 will continue reading from the teletype untill you type a ET, INSRT, or APPEND command.

```
TJ6↑/
NNTJ6 11.1
_TEST /T
>.DS
>.BEGIN 7
```

>.ET

Although the following commands can appear in source files, they are usually used with the /T command switch.

ET

End teletype input, and take input from the file. This is used to end the scope of the /T console command switch.

BEGIN <page number>

Begin output with page number <page number>. (This means page number n, not the nth page. For instance if your text starts with page number 2, and you use BEGIN 3, the output starts with page number 3, the second page, not page number 4, the third page.

NOWAIT BZ

Don't wait for a space to be typed on input console at start of each output page. (Default for output to DSK, LPT, TPL, AI, ML, DM.)

Error Messages

Each error message is followed by the name of the offending input file, the page and line numbers in that file, and the chapter, if any, page, and line numbers in the output file, e.g.,
DSK:FONTS;UGLY KST FILE NOT FOUND
Can't read font DSK:JDC;T 1 (1,2) (1,0)

When the error is caused by a command, TJ6 sometimes also prints the name of the offending command.

>7 nested inserts
TJ6 will ignore the attempted .INSRT.

Adjust bug
TJ6 adjusting bug.

Backspaced over space
TJ6 outputs a line whenever it comes to the end of a word, and there is enough stuff to output. The end of a word is a space, tab or carriage return. If you backspace past a space, you run the risk of backspacing to stuff already output by TJ6. This is a warning message indicating that you backspaced over a space, but not into stuff already output.

Backspaced too far
You backspaced past the beginning of a line, or tried to backspace into text already output by TJ6 (see above). The backspace will be ignored.

Bad file name format
Blank file name, more than one ; or : in a file name, or too many _s.

Bad font #
Number other than 0-5 following .FONT, €, or .SELECT; or non-number following €.

Bad font file
Something was wrong with the internal format of the font you designated.

Bad GENNUM counter
Second argument for GENNUM or GENSET was <1 or >8.

Can't read font
TJ6 couldn't open a font file. This message follows one naming the file and giving the ERR device reason for the failing .OPEN.

command must appear before text
The following commands must appear before text, blank lines, or commands which cause input or output (like ^, ∞, GENNUM, SPACE, PAGE, or FIGURE): XGP, FONT, SQUISH.

field too wide
The argument for HEADER was more than 120 columns wide, or the argument for HE1 or HE2 was wider than 60 columns.

Free storage exceeded
TJ6 has exceeded its free storage space. Probably due to a TJ6 bug, but might indicate excessive overprinting.

Illegal font name
Font file device name was neither DSK nor AI.

Input

TJ6 couldn't open input file. Followed by ERR device message indicating why.

Line length<1

Line length is less than or equal to zero. I.E., number of indentations plus number of leading spaces is greater or equal to TW.

Line too long

A NOFILL line, or a non-NOFILL word was longer than 127 characters. Only 127 characters will be printed.

line wider than TW

A line in NOFILL mode or a CENTER or RIGHT line sticks out of the text area. It will be printed anyhow.

Non 6-bit character in file name

File names are all sixbit.

Non-numeric argument

Command that expected a numeric argument got a non-numeric one.

Output

TJ6 couldn't open output file. Followed by ERR device message indicating why.

pages

Not an error. Message printed by TJ6 indicating the total number of pages printed.

TJ6BUG

Something is rotten in the state of TJ6. Should be followed by a more explict message and/or bug location.

TL>PL

When using SINCH, you tried to make TLINCH>PLINCH, or when a FONT command was given, TL>PL.

Too many eol calls

A TJ6 bug indicating one of its buffers has overflowed. The last command will be ignored.

Too many leading spaces

Number of spaces and/or tabs at the beginning of a line is ≥TW. The line will have mod TW spaces in front of it.

Too much GENTXT

A bug indicating that a buffer has overflowed due to a text generating command like GENNUM, ∞, π, or ∧. Part of the generated text will be lost.

Too much script

When a ↑ or ↓ in the input text, total amount of superscript is ≥64, or total subscript >64.

TW too wide

If not using the XGP, TW>127. TW will be set to 127.

TW>PW

When using SINCH, you tried to make TWINCH>PWINCH; or when a line was output or a FONT command was given, TW>PW.

unknown command

Follows the name of a command that TJ6 didn't recognize. The rest of the command line will be lost.

Historical Note

TJ6 was written in the late 1960's by Richard Greenblatt to facilitate the writing of his paper on the chess program, and has since been "improved" by almost every system programmer, and many other ITS users. This memo replaces one written by Greenblatt, Horn, and Krakauer, (AI Memo 164A, The Text-Justifier TJ6, Greenblatt, Horn, Krakauer, June 1970,) which in turn replaced the first TJ6 memo, (AI Memo 164, Producing Memos Using TJ6, TECO, and the Type 37 Teletype, Krakauer, September, 1968).

Many of the changes introduced to TJ6 have been to satisfy the whim of some programmer for some formatting trick in his memo, but the major revisions represent a history of the output hardware of the AI Lab. As seen by the title of the first memo, TJ6 was first used to output to a model 37 Teletype. The second memo was contemporary to our IBM Selectrics and had an appendix entitled "The 'Selectric' output device". Another appendix on "Inserting lower case letters into the TECO buffer" was due to the lack of lower case terminals, and the need for a trick to get lower case into the TECO buffer from our model 33 Teletypes and GE "bagbiters" (Datanet 760's).

The latest round of changes, made by the author, is aimed mostly at the Xerox Graphic Printer. TJ6 knows about variable width fonts, underlining, and other XGP features, and much of the code relating to Selectrics has been removed. I have also spent much time trying to straighten out some of the internal workings of the program, and trying to make TJ6 a bit more human to use (by adding new features, error messages, and defaults.)

Another round of changes has been aimed at simplifying and speeding up TJ6, largely by reducing it in size. This effort has been rewarded. TJ6 has been reduced from about 14 to 6 pure pages, in spite of the addition of much code needed to handle variable width fonts and the XGP.

Although everyone probably has a favorite feature he or she would like added to TJ6, the program will probably remain frozen. This freeze is a reaction to the sorry state that TJ6 was once in, when everyone's favorite feature of the week was added, and it became hard to run TJ6 without exciting some sort of bug.

Making changes to TJ6 is ordinarily difficult because of complicated interactions among variables. This difficulty, plus the desire to keep TJ6 small and fast militate against most changes. Although some features will doubtless be added to TJ6, some of the most frequently requested changes: footnotes, indices, etc., are absolutely out the question, as they would require total rewriting of TJ6 to be a two pass compiler like PUB. If that sort of feature is important to you, use PUB.

Bugs

I am glad to fix any TJ6 bugs, but I need a copy (or pointer to) the source file that produces them. A bug which seems as though it could be easily reproduced is often caused by some complicated interaction of commands and text in the source file. It's frustrating to struggle to reproduce such bugs, when someone should have saved me the trouble by sending me a file that is know to do so.

Overprinting

Simple overprinting works correctly in TJ6, but overprinting coupled with underlines, font shifts, or super- or subscripts will probably not give you the desired output. This is because of TJ6's internal line storage format. TJ6 stores lines column by column, with each column stored as a linked list of characters. When a printing character is read, it is CONSed onto the end of the current column, and the column number is incremented. Non-printing characters (∈n, ↑, ↓, ↔, →, ←,) are CONSed to the end of the column list, but the column number is not incremented. Backspacing simply decrements the column number. TJ6 then prints the line column by column, centering all the characters in each column.

The input string:

A<backspace>B<backspace>CD

is stored internally as

Column 1: A B C
Column 2: D

and will be printed correctly. The input string

e€3<backspace>^€0tre

is stored as

Column 1: e ^
Column 2: €3 €0 t
Column 3: r
Column 4: e

Since all the characters in column 1 are printed before those in column 2, the intended effect will be lost. However,

e<backspace>€3^€0tre

is stored as:

Column 1: e €3 ^
Column 2: €0 t
Column 3: r
Column 4: e

and prints correctly as

ötre.

(Since the internal line storage format is convenient for text justification, and the combination of overstriking and font switching is rarely used, it hardly seems worth the trouble to change the internal format.)

Sources

Most of the strange quotations in this memo are from Koestler's The Lotus and the Robot, and Berman's Talks on American Law.

Files

Some of the ITS files related to TJ6 are:

TJ6;TJ6 >	source for TJ6
SYSBIN;TJ6 BIN	assembled unpurified binary
SYS;TS TJ6	purified binary (This is what actually runs.)
SYS;TS OTJ6	antique version of TJ6
TJ6;TJ6MEM >	source for this memo
TJ6;TJ6MEM CONTEN	contents for this memo
TJ6;ERROR >	error messages for this memo
TJ6;TJ6MEM INDEX	index of this memo
TJ6;RECENT >	recent changes to TJ6
TJ6;TJTST >	TJ6 test file, INSRTs:
TJ6;SPTEST >	INSRTed test file
TJ6;TABTST >	INSRTed test file

Sample Input

```
.fill
.center
→Simple Sample←
.SP
Here is a simple sample of some input and output intended primarily
for people who have never used TJ6.
The TJ6 input file is a mixture of text and command lines.
Command lines start with a period and tell TJ6 what to do with the
text.  These lines are filled -- the left edges are aligned, and
the right edges don't go beyond the margin.
.sp
.adjust
These lines are adjusted -- both margins are exactly aligned.
In order to use TJ6, you must start the program from DDT, and
give TJ6 the name of an output and source file, separated by a _.
For instance,  (text generated by the computer is →underlined←)
.nf
.in 8
:TJ6 <carriage return>
→NNTJ6 11.1
_← TEST OUTPUT_TEST INPUT <carriage return>
.INDENT 0
.ADJUST
If you don't give TJ6 an output file name, it will use the input first name
and MEMO, and if you don't give it an input second name it will use >.
For instance,  the following lines have the same effect:
.nf
            :TJ6 TEST
            :TJ6 TEST MEMO_TEST >
```

Simple Sample

Here is a simple sample of some input and output intended primarily for people who have never used TJ6. The TJ6 input file is a mixture of text and command lines. Command lines start with a period and tell TJ6 what to do with the text. These lines are filled -- the left edges are aligned, and the right edges don't go beyond the margin.

These lines are adjusted -- both margins are exactly aligned. In order to use TJ6, you must start the program from DDT, and give TJ6 the name of an output and source file, seperated by a __. For instance, (text generated by the computer is underlined)

```
            :TJ6 <carriage return>
            NNTJ6 11.1
            __ TEST OUTPUT_TEST INPUT <carriage return>
```

If you don't give TJ6 an output file name, it will use the input first name and MEMO, and if you don't give it an input second name it will use >. For instance, the following lines have the same effect:

```
            :TJ6 TEST
            :TJ6 TEST MEMO_TEST >
```

Index

Note that this index is simply the TJ6 produced table of contents, as modified by the TECO alphabetic sort command. Section Titles are underlined.